Assignment 1: Document Collection Processing

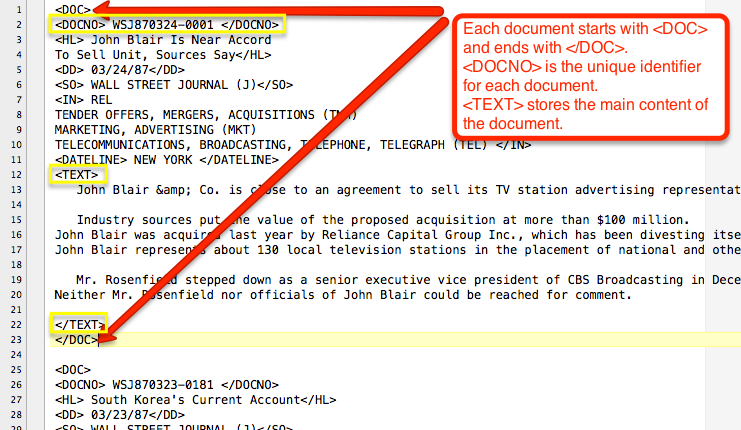
**Instruction**

The goal of the assignment is to develop Java classes that can process TREC standard format document collections. You need to follow the instructions to implement the required classes for processing the document collection files.

In information retrieval (IR) experiments, a document is usually the basic unit to be indexed and retrieved, such as a webpage or a pure text file. To simply the processing of documents, we usually compile small document files into large collection files. Each single collection file contains the contents of many small documents. trectext and trecweb are two popular document collection formats. In this assignment, we need to write programs to process documents stored in these two types of collections.

The trectext format is widely used to store textual documents such as news articles. Figure 1 shows a sample snippet of a trectext format collection file: each document starts with <DOC> and ends with </DOC>; <DOCNO> stores the unique identifier of each document; <TEXT> stores the main content of the document.

The trecweb format is used to store web documents. Figure 2 shows a sample snippet of a trecweb format collection file: each document starts with <DOC> and ends with </DOC>; <DOCNO> stores the unique identifier of each document; <DOCHDR> usually stores the http response header information of accessing the web document; the content of the web document is stored between </DOCHDR> and </DOC>.



**Figure 1: A sample snippet of a trectext format collection file.**



**Figure 2: A sample snippet of a trecweb format collection file.**

**Tasks**

### Task 1: Reading Documents from Collection Files

In this task, you should implement two classes that can read individual documents from trectext and trecweb format collection files (you can find the classes and detailed descriptions in src.zip):

* **PreProcessData.DocumentCollection** is a general interface for sequentially reading documents from collection files
* **PreProcessData.TrectextCollection** is the class for trectext format
* **PreProcessData.TrecwebCollection** is the class for trecweb format

### Task 2: Normalize Document Texts

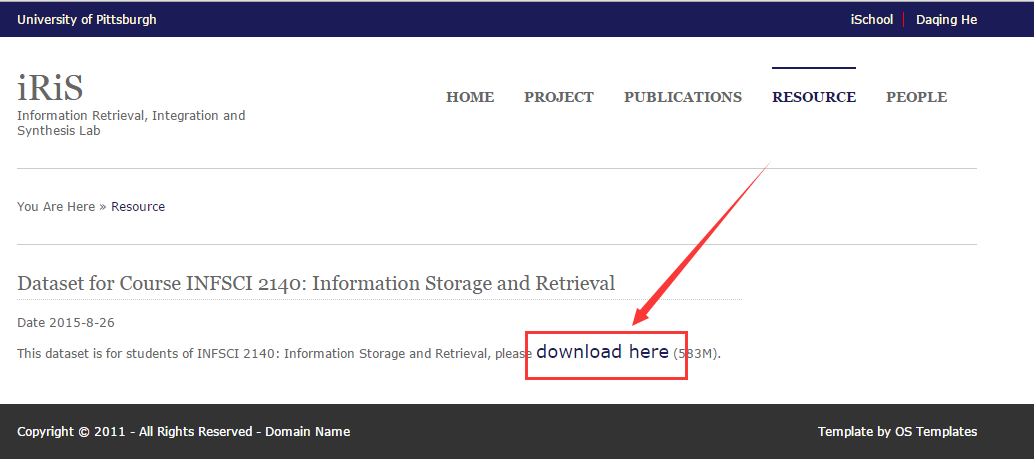
In this task, you should first implement classes to tokenize document texts into individual words, normalize all the words into their lowercase characters, and finally filter stop words.

* **PreProcessData.TextTokenizer** is a class for sequentially reading words from a sequence of characters
* **PreProcessData.TextNormalizer** is the class that transform each word to its lowercase version, and conduct stemming on each word.
* **PreProcessData.StopwordsRemover** is the class that can recognize whether a word is a stop word or not. A stop word list file will be provided, so that the class should take the stop word list file as input.

The 6 classes to be implemented in task1 & 2 can be found in src.zip. You **CANNOT** change the classes’ names or the required methods’ names. However, you can add new variables, constants, and methods in these classes and create new classes if necessary.

**HW1Main** is the main class for running your assignment 1. You can find the class in src.zip, and you are **NOT** allowed to change anything in this file. If you have successfully implemented the 6 classes in task 1&2, you should be able to directly run HW1Main, which can read an input collection file and output the normalized version of each document in the collection as another file.

**Classes.Path** contains addresses of all input and output files, so you should put all files in the corresponding directory. **Classes.Stemmer** is the stemmer that you will use to normalize each word, and you can learn how to use this stemmer through its main method. These two classes are **NOT** allowed to change, too.



**Figure 3: Download the collection files.**

Two collection files and stopword.txt are provided in <http://crystal.exp.sis.pitt.edu:8080/iris/resource.jsp> , as shown in Figure 3. docset.trectext is a trectext format file and docset.trecweb is a trecweb format one. Your programs should at least be able to successfully process these two documents.

**Requirements and Reminders**

* You CAN ONLY use Java and Python to finish this assignment.
* You CANNOT use any Java API library other than the standard JDK (for example, you cannot use apache commons, apache Lucene, indri etc. in this assignment).
* Feel free to use IDE tools such as Eclipse, Netbeans, and Spyder.
* Do tell us the Java version you used for writing your assignment, e.g. JDK 1.6 or 1.7. Please only use Oracle JDK or openjdk.

**Grading**

Your submission will be graded based on:

* Correctness of the implementation on the required functions (70%)
* Efficiency of your implementation, make sure your code finish processing two collections within 15 minutes (20%)
* Necessary program annotation and commentaries (10%)

## Submission Requirements

A zipped file package with the naming convention as “pittids\_2140a1”. For example, suppose the Pitt id is jud1, then the submission package should be jud1\_2140a1.zip.

The file package should contain:

1. All the scripts/programs you used for this assignment (**src folder**)
2. A short instruction on how to read your scripts and how to run your scripts, including environment configuration. (This should be a **txt file**.)
3. A short instruction on how long it takes to finish running your code, and how many documents of two corpus are processed. You will **NOT** upload your result file, so make sure your code finish running within 15 minutes; if cannot, you must make it clear whether the code will generate the result, and how much time it costs to finish running. (This should be a **txt file**.)